IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: Heeloo Chung, et al.	§	Docket No.:	35399.42
	§	Customer No.	27683
Serial No.: 09/990,753	§		
	§	Group Art Unit:	2663
Filed: November 16, 2001	§		
	§	Examiner:	Derrick W. Ferris
For: High Efficiency Data Buffering In a Computer Network Device	§		

SECOND DECLARATION UNDER 37 C.F.R. § 1.131

I, Heeloo Chung, declare and say that:

1. I am a joint inventor of the subject matter disclosed and claimed in the above-identified application.

2. At all times set forth herein, I was employed at Force10 Networks, Inc. ("Force10"), Milpitas, California. Force10 is the assignee of the above-identified application.

3. Prior to July 18, 2001, I was part of a Force10 team who reduced the subject matter of the present invention to practice in Milpitas, California, in an Application Specific Integrated Circuit (ASIC) we designed, codenamed "Cougar."

4. Exhibit A contains 17 pages excerpted from a Cougar ASIC Internal Design Specification, numbered manually for reference in this declaration. I was one of the authors of this document. Although minor details of the Specification regarding control register definitions not important to this invention were updated just subsequent to July 18, 2001, the subject matter appearing on the pages excerpted in Exhibit A was placed in the Specification prior to July 18, 2001. Sensitive material not pertinent to this declaration has been redacted from pages 16 and 17 of Exhibit A.

5. Pages 1-3 of Exhibit A show the general architecture of a network device (the "Cyclone switch system") with the Cougar ASIC serving as a line card buffer manager between an ingress/egress ASIC (codenamed "Tiger") and a switch fabric comprising switch fabric ASICs (codenamed "Simba"), with switch fabric transfers controlled by a scheduler ASIC (codenamed "Panda").

6. The Tiger ingress/egress ASIC performed forwarding lookup for packets received by that Tiger, assisted by another ASIC (codenamed "Cheetah") that maintained and searched forwarding databases. The forwarding lookup allowed the Tiger to send the Cougar ASIC packets containing an "F10 header"—a proprietary header that was prepended to each packet prior to passing the packet to Cougar. The Cheetah ASIC is referred to on pages 16-17 of Exhibit A as a source of header information for the Cougar ASIC.

7. Pages 16-17 of Exhibit A show the format of the F10 header. Some details of the F10 header format that are not relevant to this declaration have been redacted. What is shown is that bits 23:16 of the F10 header contain a description of the Switch Fabric Destination Port ID (SFDP-ID) for a packet, as obtained from a Cheetah lookup table, and bits 26:24 describe a Queue Pointer (QP) for one of eight traffic classes, also obtained from Cheetah. The SFDP-ID and QP associate each packet with a particular Virtual Output Queue (VOQ) maintained by Cougar, and are passed to Cougar over the Tiger to Cougar C-Port Interface in the F10 header prepended to the packet.

8. Pages 4-5 of Exhibit A show block level I/O's of the packet processing unit on the Cougar ASIC, including outputs labeled "pp_mq_(label)" that pass from the packet processing unit to the memory management unit (see also the figure on page 4.) One of these outputs, taken from the F10 header received from Tiger, is pp_mq_qid[7:0], an eight-bit queue ID that is a concatenation of five bits from the F10 header SFDP-ID (only five bits are valid in this implementation) and the three bits of the QP from the F10 header. The signal pp_mq_id is the requested queue number that the memory management unit (MMU) is supposed to write the current packet to.

9. Pages 6-15 of Exhibit A show details of the buffer manager implemented on the Cougar ASIC, with data managed in "chunks" analogous to the "trunks" disclosed in the present patent application. The Cougar ASIC buffer manager implements the buffer manager features claimed in the present application.

10. The MMU managed up to 256 unicast virtual output queues (32 switch fabric ports, 8 classes per port) and 128 multicast virtual output queues, as described on page 7 of Exhibit A, writing packets to these queues based on pp_mq_qid[7:0] received from the packet processing unit.

11. The MMU stored data packets in link-listed chunks, using a head pointer and tail pointer for each virtual output queue. Pages 11-15 of Exhibit A describe the queue structure, including an example with five packets p1, p2, p3, p4, and p5 stored in queues 0, 1, and 383 (a multicast queue). Packet p1 includes three chunks p1a, p2a, and p3a, illustrating how a packet with greater than two chunks is stored in queue memory. Packet p4 includes two chunks p4a, p4b, illustrating how a packet with two chunks is stored in queue memory. Packets p2, p3, and p5 illustrate one-chunk packets.

12. As I stated in my previous declaration, I am confident that we had reduced the Cougar ASIC to practice, and a network device containing Cougar ASICs to practice, prior to July 18, 2001. The Cougar ASIC included all of the functionality I have described in this declaration.

I declare that all statements made herein of my knowledge are true and that all statements made on information and belief are believed to be true and that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code.

Heeloo Chung

Date: 8-1-2006

3 Cougar(BDB) System Level Specification

3.1 The Architecture Overview

The Cougar is a buffer/queue manager ASIC in the Cyclone switch system. It consists 7 major blocks, Buffer/Queue Management Unit (MMU), Packet Processing Unit (PPU), Output Organizer Unit (OU), Front End C-Port Interface (FC), Backplane C-Port Interface (BC), CPU Interface (PCI), and Panda Scheduler Interface (PI). The following figures illustrate the data flow in the Cougar architecture.

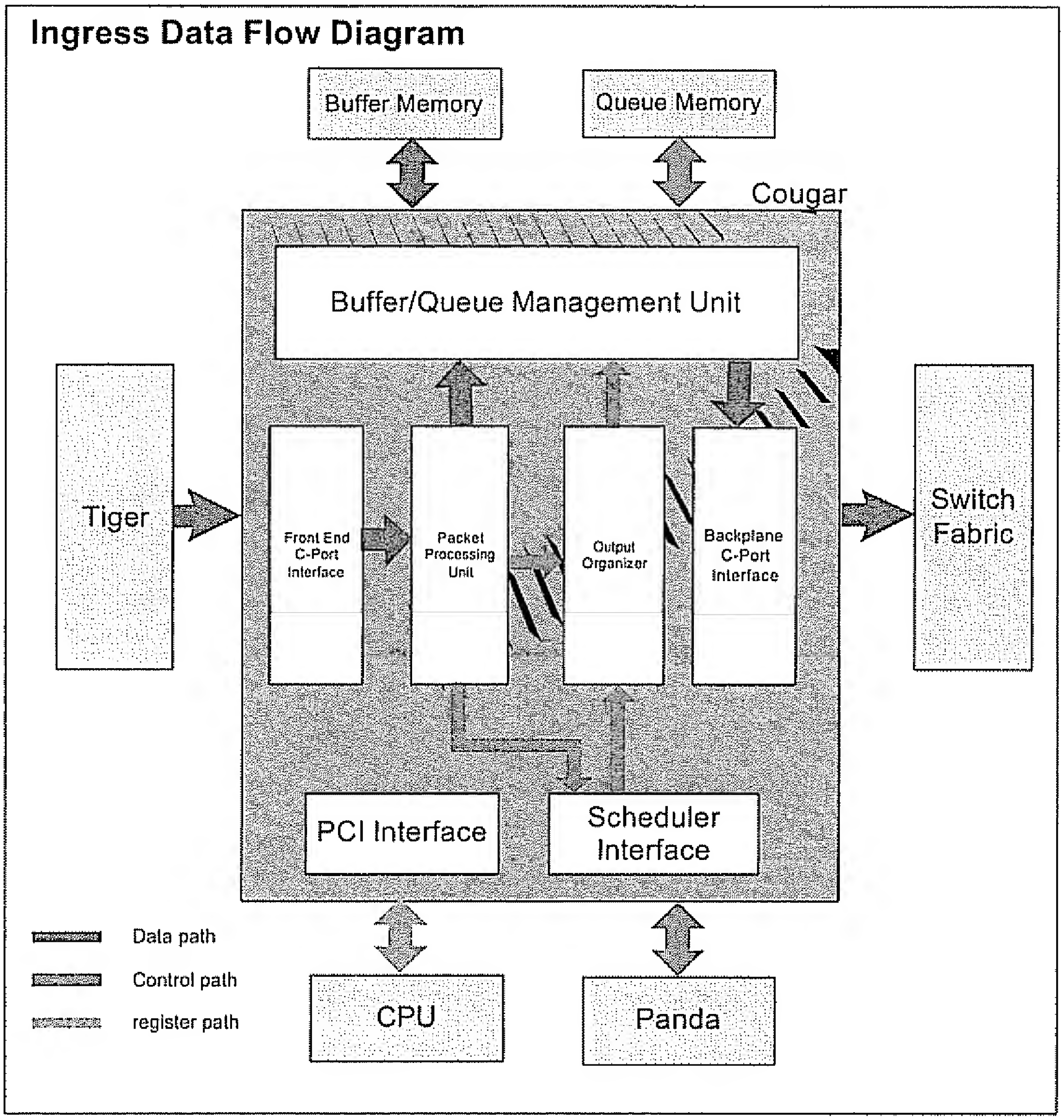


Exhibit A
Page 1

Egress Data Flow Diagram

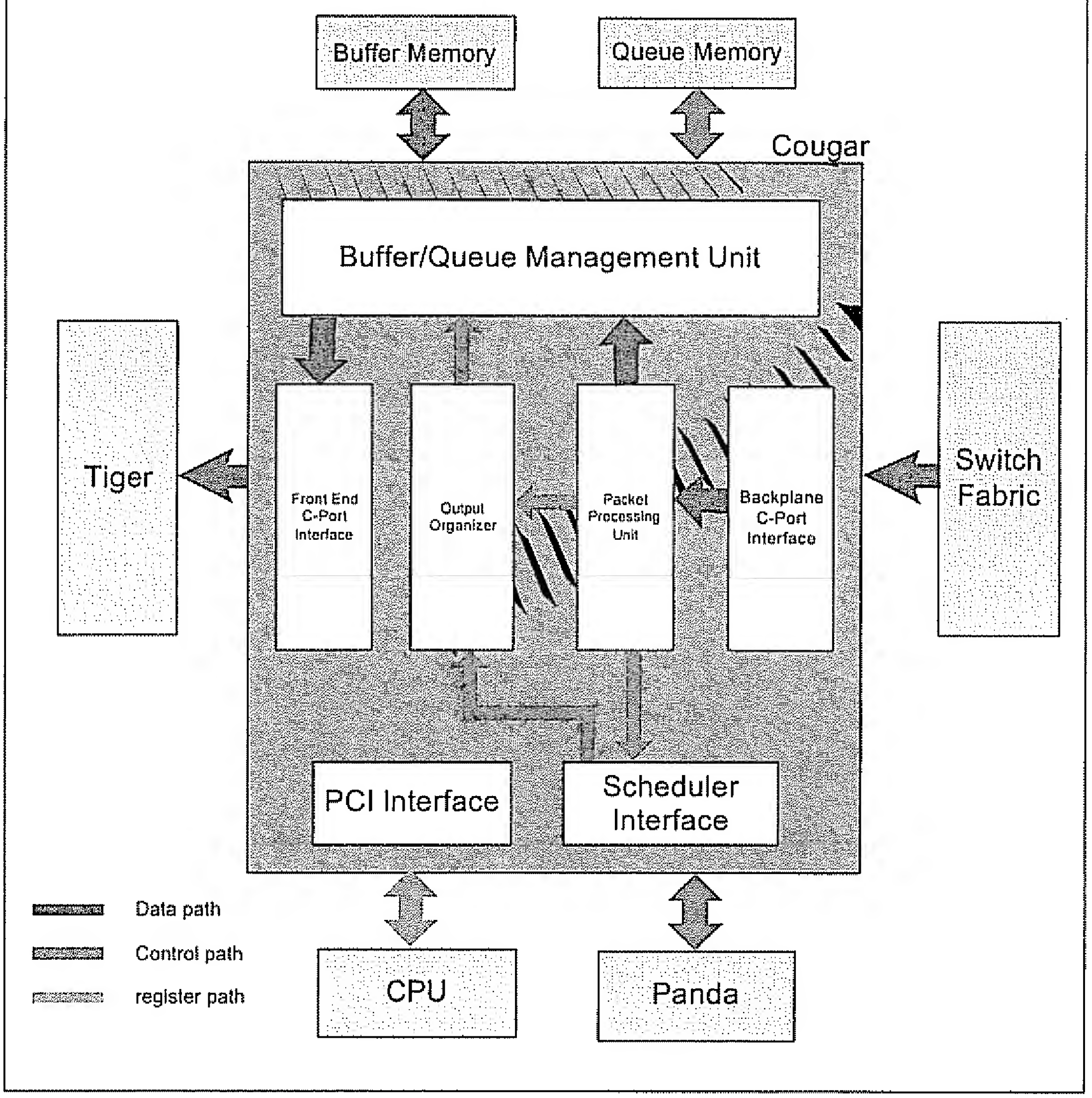
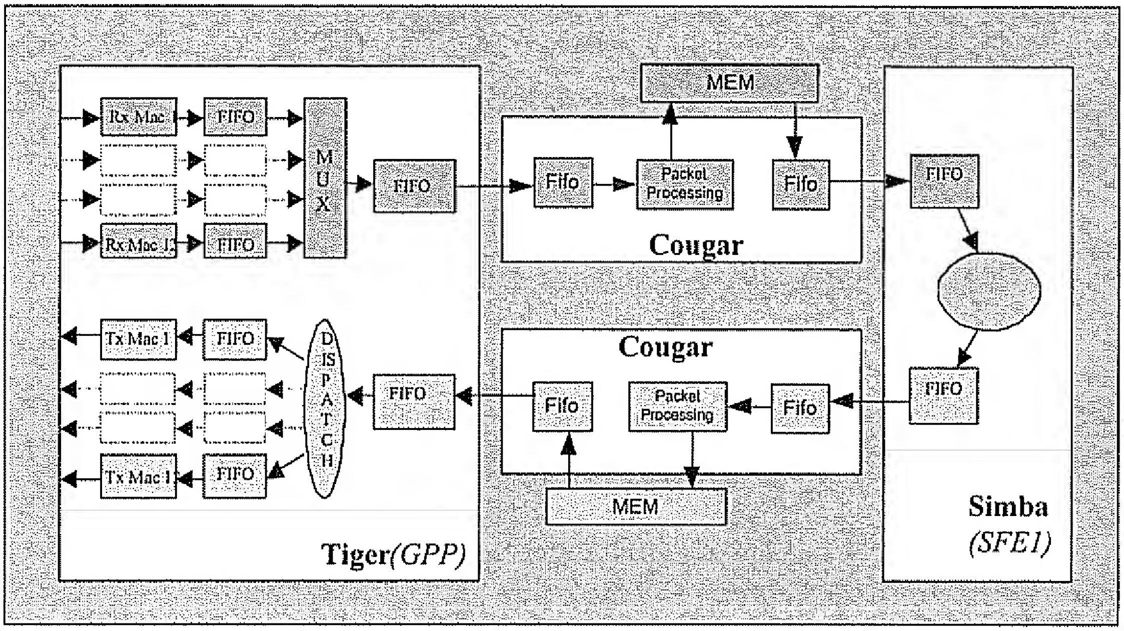


Exhibit A
Page 2

3.2 System level Data Flow

Cougar chip is used multiple times in the Cyclone system. It is used in both the receive data path and transmit data path. For each Tiger used in the system, two Cougar will be used, one for receive and one for transmit. A data packet comes into the system will go through two Cougars. See the following basic data flow diagram. In the current design, each Cougar can handle up to 12 Gbps of input data traffic.



3.3 Chip Level Interfaces

Cougar has 6 external interfaces, one Tiger C-port interface, one back plane C-port interface, two memory interfaces, one CPU interface, and one scheduler interface. The Tiger C-port interface is used in between the Tiger and Cougar. The back plane C-port interface is in between the Cougar and the back plane switch fabric. Two memory interfaces are used for data storing and forwarding. Panda is the scheduler for the back plan switch fabric. Cougar will use the scheduler interface to communicate it. During the system power up, PCI interface will be used to initialize the chip. This interface can also be use for monitoring the status during the normal operation. All interfaces are described in detail in the following sections.

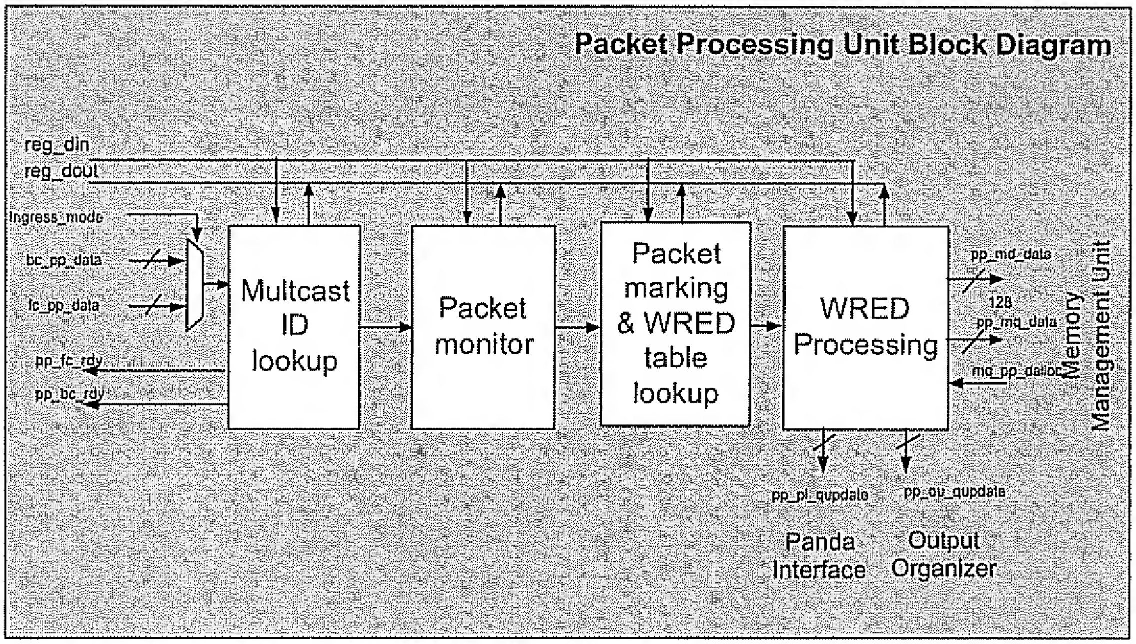
3.3.1 Tiger C-Port Interface

C-Port interface is a DDR (double data rate) bus interface, runs at 160MHz. Since each Tiger has to handle 12 ports at the same time, data packets will be multiplexed in and out of the C-port in TDM fashion. C-port has two modes. It works as input only interface in the ingress path, and output only on the egress path. Most of signals are bi-directional I/O's. It consists of the following signals:

Exhibit A
Page 3

4.2 Packet Processing Unit

Cougar provides a set of Differentiated Service (Diffserv or DS) features on a per packet basis. In this unit, each incoming IP packet is monitored at the input port, and processed with the WRED mechanism before stored into the buffer memory. See the following block diagram.



4.2.1 Block level IO's

	Signal	I/O	Description
global	cclk	I	core clock 160 Mhz
	core_rstn	I	core reset
	ingress_mode	I	Ingress mode select from pin
fc signals	fc_pp_data[127:0]	I	128-bit Data bus
	fc_pp_data_vld	I	Valid data
	fc_pp_head	I	Packet header indicator
	fc_pp_tail	I	Packet tail indicator
	pp_fc_rdy	O	PPU ready to accept a data
md signals	pp_md_data[127:0]	O	Write data to memory
	pp_md_header	O	Packet header indicator
	pp_md_tail	O	Packet tail indicator
	pp_md_valid	O	PPU data valid
	md_pp_free[6:0]	I	number of free entries in the FIFO
mq	pp_mq_valid	O	PPU data valid
	pp_mq_header	O	packet header

Exhibit A
Page 4

	Signal	I/O	Description
	pp_mq_tail	O	packet tail
	pp_mq_qid[7:0]	O	QID = (sfpid[4:0],cid[2:0])
	pp_mq_psize[6:0]	O	Packet size in 16Byte blocks
	pp_mq_mcast	O	multicast packet
	mq_pp_ready	I	MQ is ready to accept data from PPU
	mq_ppi_qid[7:0]	I	queue de-allocate update QID
	mq_ppi_dalloc	I	queue de-allocation request
	mq_ppi_mcast	I	multicast queue de-allocate
	mq_ppi_size[6:0]	I	packet size in 16byte blocks for de-allocation after ceiling
ou signals	pp_ou_mc_cnt_en	O	multicast count enable
	pp_ou_mc_gid[7:0]	O	multicast group ID
	pp_ou_qupdate	O	unicast queue update
	pp_ou_qid[7:0]	O	unicast queue ID = (sfpid[4:0],cid[2:0])
	pp_ou_empty_flag	O	unicast queue empty flag: 1 for empty, 0 for not empty
pi	pp_pi_size_vld	O	ppu to pi size valid
	pp_pi_size[6:0]	O	packet size in 16-byte blocks
	pp_pi_dpid[4:0]	O	switch fabric dest port ID
bc signals	bc_pp_data_vld	I	bc data to ppu is valid
	bc_pp_data[127:0]	I	bc data bus to ppu
	bc_pp_header	I	bc packet header indicator
	bc_pp_tail	I	bc packet tail indicator
	pp_bc_rdy	O	ppu ready to accept next data from bc
PCI interface	reg_din[31:0]	I	register data in bus from previous block
	reg_adr_in[15:2]	I	register address bus from previous block
	reg_req_in	I	register access request from previous block
	reg_wr_in	I	register write in from previous block
	reg_dout[31:0]	O	read/write data out to next block
	reg_adr_out[15:2]	O	address out to the next block
	reg_wr_out	O	register write command out to the next block
	reg_req_out	O	register request out to the next block

4.2.2 Multicast ID lookup

Multicast traffic is handled quite differently in ingress Cougar and egress Cougar. In ingress Cougar, the 3-bit super group ID is used to store the multicast packets into 8 queues, then multicast (or broadcast) out to the egress cougars. This ID lookup block is in a pass through mode in ingress Cougar. In egress Cougar, all the incoming multicast packets will have to go through this ID lookup to obtain the queue ID and be stored into 128 queues in the buffer memory, then they will be converted into unicast packets before transmitting to Tiger. However, each egress cougar may receive some multicast packets, which don't belong to it. So it will drop those packets after the ID lookup. The lookup table is 4Kx8 table. It can support up to 4K multicast groups. In the 8-bit entry, the MSB

4.4 Memory Management Unit

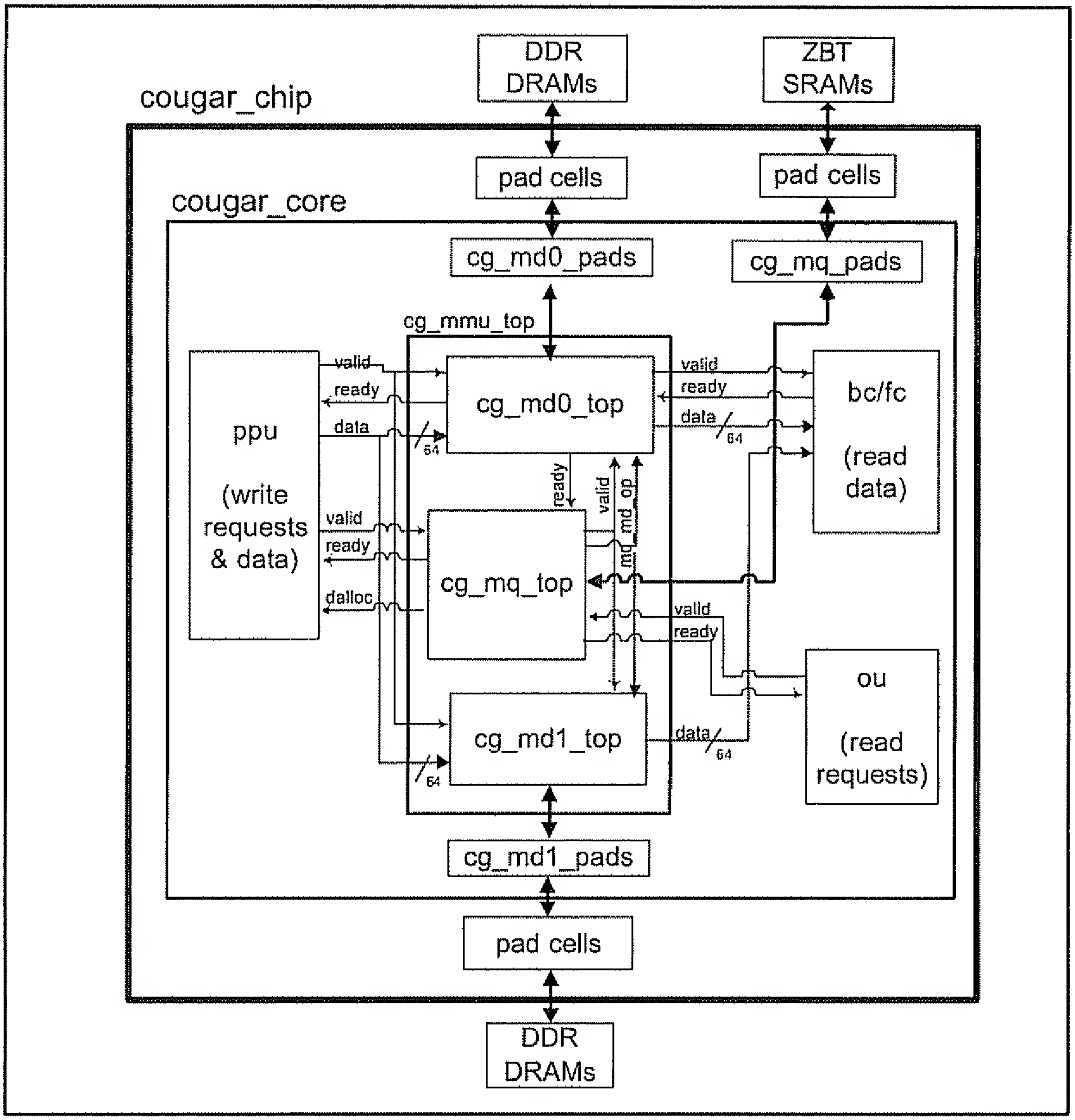


Exhibit A
Page 6

4.4.1 Overview

4.4.1.1 Terminology

• Chunk	128-byte data storage unit. Corresponds to a burst of 4 words to/from the DDR DRAMs. A chunk is made of 8 blocks.
• Block	16-byte data unit written by the ppu or read by the fc or bc.
• Template	A sequence of commands to the DDR DRAMs that accesses four different banks. Made of four slots.
• Slot	Access to a single bank in a template
• ECC	Error correcting code
• Header ECC	The ECC bits for the 96 bit (12 byte) F10 header in data memory. Occupies 8 bits.
• Pointer	A single location in external queue SRAM that contains a link to the next node, the packet length and ECC bits.

4.4.1.2 Structure

The mmu block consists of six core-level blocks, as shown on the diagram above. The blocks are:

1. cg_mq_top — Receives write requests from the ppu and read requests from the ou. Groups the reads and writes into DRAM templates with two reads and two writes. Allocates and deallocates the memory chunks from the freelists. Maintains FIFO head and tail information for up to 256 unicast and 128 multicast queues.

2. cg_mq_pads — Registers that are placed close to the ZBT SRAM I/O cells on the layout. Includes the ECC generator and checker for the pointers.

3. cg_md0_top — The data unit for one half of the 256 DDR DRAM pins. Contains read and write FIFOs and a DRAM controller sequencer. Also contains a header ECC checker/corrector for half the DRAM pins.

4. cg_md1_top — The data unit for the second half of the 256 DDR DRAM pins. Contains read and write FIFOs and a DRAM controller sequencer. Also contains a header ECC checker/corrector for half the DRAM pins.
5. cg_md0_pads — Registers that are placed close to the DDR DRAM I/O cells on the layout. Includes the DDR read data strobe delay lines and FIFOs.
6. cg_md1_pads — Registers that are placed close to the DDR DRAM I/O cells on the layout. Includes the DDR read data strobe delay lines and FIFOs.

4.4.3 Queue Unit

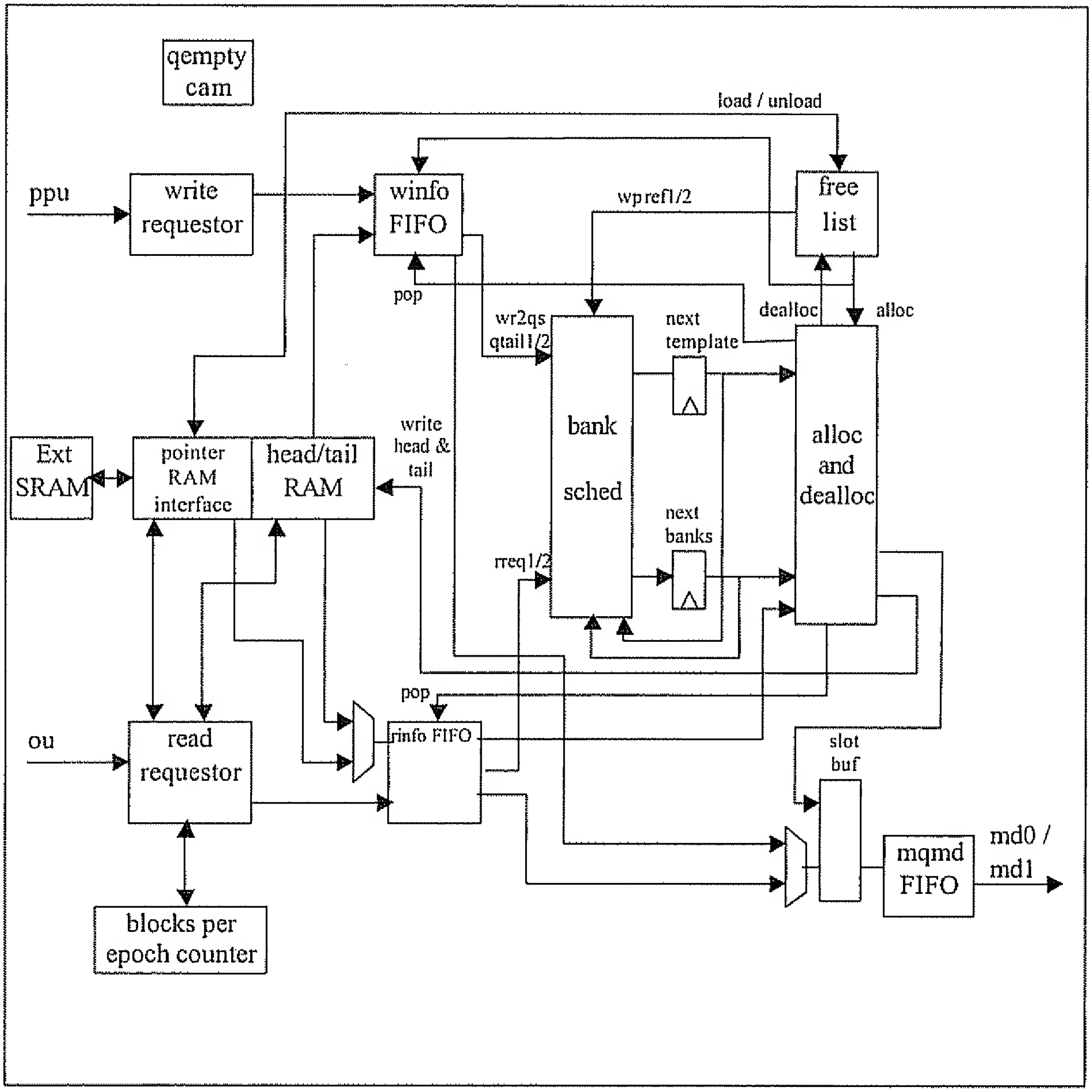
4.4.3.1 Block level IO's – cg_mq_top

	Signal	I/O	Description
gbl sig	cclk	I	core clock 160 Mhz
	core_rstn	I	core reset
	ingress_mode	I	1=ingress Cougar, 0=egress Cougar mode
ppu signals	pp_mq_valid	I	PPU data valid
	pp_mq_header	I	packet header
	pp_mq_tail	I	packet tail
	pp_mq_qid[7:0]	I	QID = (sfpid[4:0],cid[2:0])
	pp_mq_psize[6:0]	I	Packet size in 16Byte blocks
	pp_mq_mcast	I	multicast packet
	mq_pp_ready	O	MQ is ready to accept data from PPU
	mq_ppi_qid[7:0]	O	queue de-allocate update QID
	mq_ppi_dalloc	O	queue de-allocation request, this signal can only active in N clock cycles, (N>4)
	mq_ppi_mcast	O	multicast queue de-allocate
	mq_ppi_size[6:0]	O	packet size in 16byte blocks for de-allocation after ceiling
ou signals	ou_mq_valid	I	read request
	ou_mq_qid[7:0]	I	read QID = (sfpid[4:0],cid[2:0])
	ou_mq_mcast	I	multicast queues
	ou_mq_last_mcast	I	last multicast transfer, de-allocate enable
	ou_mq_epoc_stop	I	epoch stop, no more epoch data
	mq_ou_ready	O	MQ acknowledge
	mq_ou_qdone	O	requested queue reaches the end of queue
	mq_ou_qempty	O	requested queue is an empty queue
	mq_ou_eoe	O	end of epoch
md signals	mq_md_op[42:0]	O	memory read/write template, address, size and slot information
	mq_md_valid	O	mq to md fifo not empty
	md_mq_ready	I	mq to md fifo pop
	bc_md_ready	I	ready signal to data unit in ingress mode – NOT USED.
	fc_md_ready	I	ready signal to data unit in egress mode – NOT USED.
	md_valid	I	valid signal from the data unit – NOT USED.
Pad Signal	pad_mq_serr	I	Indicates that a single bit ECC error has been detected.
	pad_mq_syndrome[6:0]	I	ECC memory bit failures information

Exhibit A
Page 8

	Signal	I/O	Description
	mq_pad_data[35:0]	I	Pointer SRAM read data from pads including ECC bits.
	pad_mq_derr	I	ECC Double Bit Error Detected
	pad_mq_rdata_valid	I	The read return data (and Serr/Derr status) is valid.
	mq_pad_data[35:0]	O	Pointer SRAM write data to pads including ECC bits
	mq_pad_addr[21:0]	O	Pointer SRAM address and chip select
	mq_pad_read	O	Pointer SRAM read request
	mq_pad_write	O	Pointer SRAM write request
	mq_pad_ecc_dis	O	Disable the auto-generated ECC bits
	mq_pad_follow_link	O	Loop-back the current pointer data address field to the pointer address – ignore mq_pad_addr input.
PCI Sig	reg_din[31:0]	I	register data in bus from previous block
	reg_adr_in[15:2]	I	register address bus from previous block
	reg_req_in	I	register access request from previous block
	reg_wr_in	I	register write in from previous block
	reg_dout[31:0]	O	read/write data out to next block
	reg_adr_out[15:2]	O	address out to the next block
	reg_req_out	O	register request out to the next block
	reg_wr_out	O	register write command out to the next block

4.4.3.2 cg_mq_top Block Diagram



4.4.3.3 DDR DRAM Basics

DDR (Double Data Rate) DRAMs are Dynamic memory chips that are internally organized as 4 banks by R Rows by C columns. Each Column is a data word, either 16 bits or 32 bits per chip for the types that are interesting for this unit. There is an address and command bus that is sent to the DRAM synchronized by a clock. The address and command pins are single data rate (SDR). The address is sent in two parts – the row address first, then the column address.

The sequence of events needed to read from the DRAM is:

1. Wait until tRP has been satisfied for the desired bank. Send NOPs until this timing parameter is met.
2. Activate the bank - send the ACT command along with the desired bank number and row address.
3. Wait for tRCD clocks
4. Send the READAP (read with auto precharge) along with the column address and the bank number.
5. Wait until the read data strobe is returned by the DRAM.
6. Capture the read data on both positive and negative edges of the data strobe.

The sequence needed to write to the DRAM.

1. Wait until tRP has been satisfied for the desired bank. Send NOPs until this timing parameter is met.
2. Activate the bank - send the ACT command along with the desired bank number and row address.
3. Wait for tRCD clocks
4. Send the WRITEAP (write with auto precharge) along with the column address and the bank number.
5. Drive the data strobe outputs low during the preamble.
6. Drive the write data onto the DQ outputs and toggle the data strobe. The write data strobe is centered around the write data. Data is transferred on both edges of the data strobe.
7. When the entire burst has been output, drive the data strobes low during the postamble.
8. Drive the data strobes to tri-state.

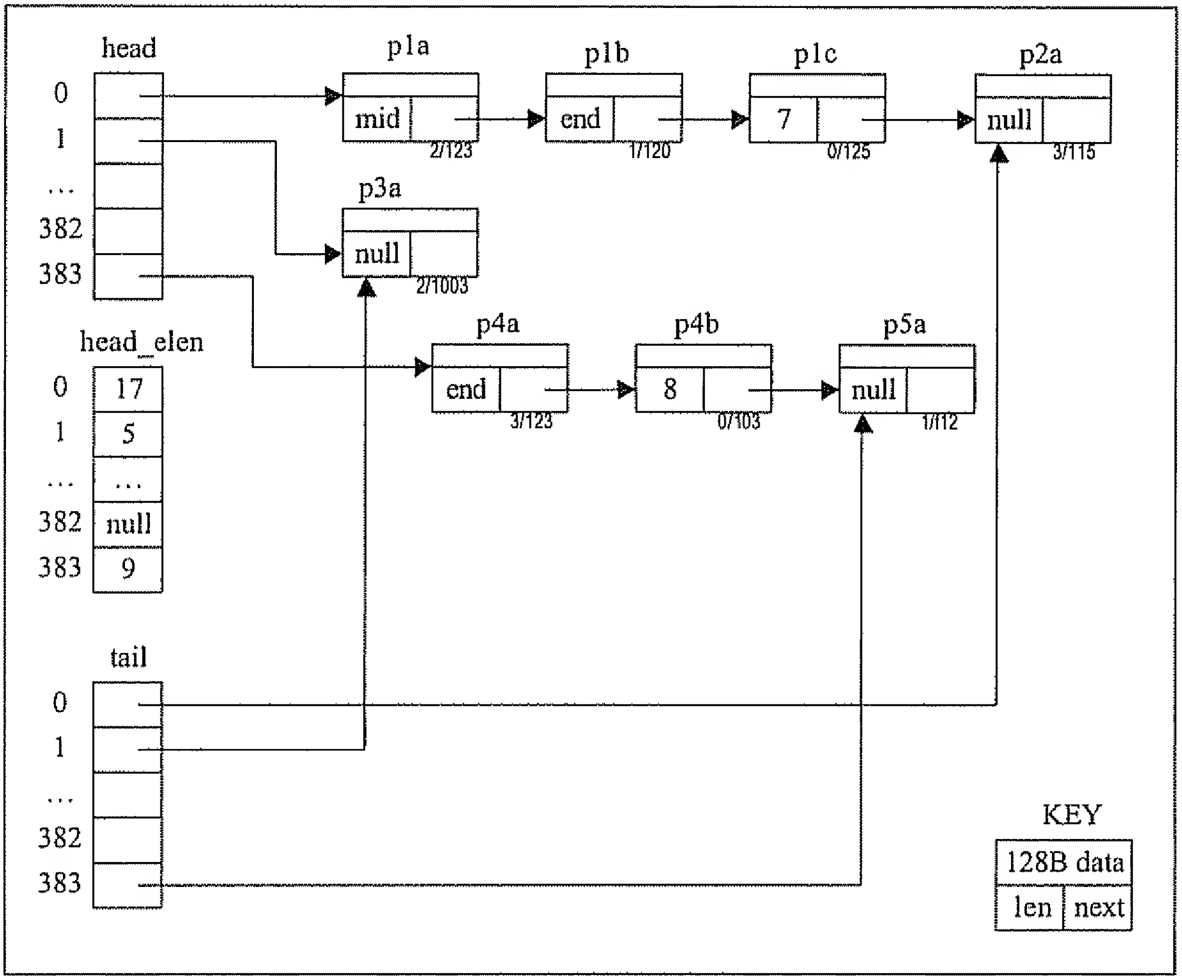
Operations for each bank can happen in parallel. We use this fact to interleave all four banks within one 13-cycle template to get two read bursts and two write bursts every 13 cycles (with best bank assignment). With worst bank assignment, up to 5 *spacer cycles* (see below) must be added, extending the template time to 18 cycles.

4.4.3.4 Queue Structure

The mq unit supports 256 unicast and 128 multicast queues. Each queue is a FIFO. For each queue there is:

- A head pointer (head[q]), which points to the first chunk in the first packet in the queue.
- A tail pointer (tail[q], which points to the last chunk in the last packet in the queue.
- A head length (head_elen[q]), which indicates the size in blocks of the first packet in the queue.

Conceptually, there are three SRAM arrays on chip that hold the queue start and end information. Physically, all three arrays are kept in a single 1-port SRAM.



4.4.3.5 Read Requests

Read requests are sent by the ou. The ou requests a queue to read by setting ou_mq_mcast, ou_mq_qid, and ou_mq_last_mcast and setting ou_mq_valid high. The mq accepts the read request by setting mq_ou_ready high. Read requests for the packets pointed to by the head[q] register for the queue will be pushed to the rinfo_fifo. The linked list for the queue is traversed until any of the following occurs to terminate reading the queue:

1. The end of the queue is reached, as indicated by reaching the node pointed to by the tmp_read_tail register. At the beginning of traversing a queue, the tail[q] value is saved to tmp_read_tail.
2. An ou_mq_epoc_stop is received and the end of a packet is seen.
3. The blocks_per_epoch counter indicates that the next packet on the list will not fit into the current epoch. This check does not happen for the first packet in the epoch.
4. If the chip is in egress mode and a multicast request is received, when the end of the first packet has been processed.

Exhibit A
Page 12

During the process of walking the linked list, each chunk is automatically deallocated (returned to the free list) except for the following two conditions:

1. multicast request and ou_mq_last_mcast was low.
2. Packet generator mode is enabled.

4.4.3.6 Write Requests

Write requests are sent by the ppu. An individual write request occurs for each chunk in the packet. Packet boundaries are indicated by the pp_mq_header and pp_mq_tail signals, which are sent along with pp_mq_mcast and pp_mq_qid when pp_mq_valid is high to initiate the request. The mq accepts the request by setting mq_pp_ready high.

For the first chunk in the packet, the size of the entire packet is indicated on pp_mq_psize, in blocks. For all subsequent chunks in the packet, pp_mq_psize should have a range from 1 to 8 blocks. The middle chunks should have a size of 8, and the last chunk a size from 1 to 8, except for one-chunk packets. One-chunk packets must have a size from 5 to 8 blocks inclusive.

The register head_elen[q] is checked for the requested queue number. If the queue is empty, then the head[q] and head_elen[q] registers will need to be written after the blocks in the packet have been allocated. If the selected queue is not empty, then the tail[q] register is read and pushed to the winfo_fifo. The qtail bank number will be used by the bank scheduler to ensure that two chunks on a single queue never have the same bank number.

All the chunks in write packet are first built off-line and linked together before being appended to the queue. The entire packet is then appended in one atomic operation. The registers tmp_write_head, tmp_write_head_elen and tmp_write_tail are used to hold the write packet while it is being built.

4.4.3.7 Bank Scheduler

The bank scheduler combines two read chunk requests with two write chunk requests to form a template to be executed by the md0/md1 units. The following templates are supported:

Template	0	1	2	3	4	5	6	7	8	9	10	11	12
RaRbWcWd	ACTa	NOP	ACTb	RDa	NOP	RDb	NOP	ACTc	NOP	ACTd	WRc	NOP	WRd
WaWbRcRd	ACTa	NOP	ACTb	WRa	NOP	WRb	ACTc	NOP	ACTd	RDc	NOP	RDd	NOP

The inputs to the bank scheduler are:

1. rreq1 & rreq2: Zero, one or two read requests – bank numbers
2. qtail1 & qtail2: Zero, one or two write requests – bank numbers for the previous nodes that the newly allocated nodes will link to.
3. wr2qs: If there are two write requests, are the requests for a single queue (need to be chained together) or for two different queues.

4. wprefl & wpref2: The bank numbers for the two freelists that have the most free entries.
5. prev1, prev2, prev3, prev4: The bank numbers for the previous template
6. prev_templ: The template type of the previous template

Spacer cycles are NOP cycles that will be inserted by md0/md1 between templates in order to ensure that the tRC, tRP and tWTR DRAM timing parameters are met. From zero to 5 spacer cycles may be needed to meet tRC/tRP. The bank scheduler will try to minimize spacer cycles by selecting the bank order and template. Note that tWTR is not considered by the bank scheduler during scheduling but is enforced by md0/md1.

The bank scheduler will output the "best" template and order of bank numbers in order to satisfy the following constraints:

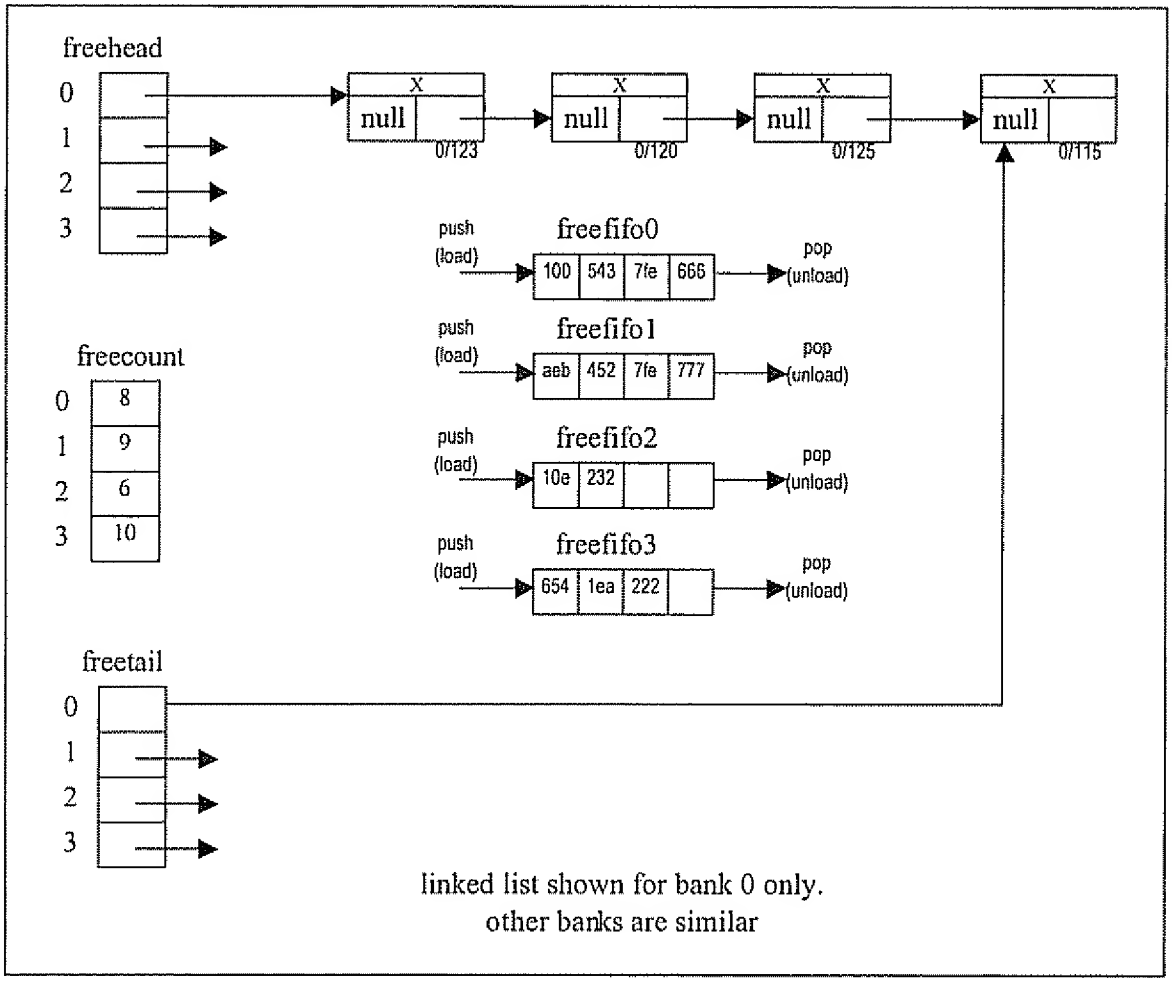
1. Never assign the same bank twice.
2. Read the requested bank(s) in the order: rreq1, rreq2
3. If rreq1 and rreq2 are to the same banks, then read rreq1 on this template and defer rreq2 until the next template.
4. Assign the remaining two banks to the write request(s)
5. Ensure that the assigned bank numbers of a write and its corresponding qtail are different – i.e. sched_wr1 != qtail1 and sched_wr2 != qtail2 if wr2qs is true; if wr2qs is false, then sched_wr1 != qtail1 and qtail2 is ignored.
6. Choose the two preferred banks, wpref1 and wpref2 if possible.
7. If there is a conflict between reads and writes in assigning banks when 2 read and 2 write requests are present, then schedule both writes but only the first of the reads, rreq1 on this template. Defer rreq2 until the next template.
8. Minimize the spacer cycles.

4.4.3.8 mqmd FIFO

The mqmd FIFO is an asynchronous FIFO that is used to send template commands from the mq module to the two md modules. The FIFO format is shown below:

Function	42-40	39	38-37	36	35-34	33	32-31	30	29-28	27-26	25-23	22-20	19-0
R1R2W3W4	100	V1	BANK1	V2	BANK2	V3	BANK3	V4	BANK4	SLOTNUM	SLOT_OP	BSIZE	ADDR_OFFS
W1W2R3R4	101	V1	BANK1	V2	BANK2	V3	BANK3	V4	BANK4	SLOTNUM	SLOT_OP	BSIZE	ADDR_OFFS
R1R2R3R4	110	V1	BANK1	V2	BANK2	V3	BANK3	V4	BANK4	SLOTNUM	SLOT_OP	BSIZE	ADDR_OFFS
W1W2W3W4	111	V1	BANK1	V2	BANK2	V3	BANK3	V4	BANK4	SLOTNUM	SLOT_OP	BSIZE	ADDR_OFFS
CONT_SLOT		V1	BANK1	V2	BANK2	V3	BANK3	V4	BANK4	SLOTNUM	SLOT_OP	BSIZE	ADDR_OFFS

4.4.3.9 Freelist Operation



4.4.3.10 End of Epoch Operation

4.4.3.11 CPU Pointer RAM Access

4.4.3.12 CPU DRAM Access

The cpu may read and write any data in DDR DRAM at any time. The address is specified by writes to two index registers – dbuf_windex and dbuf_rindex. The data is accessed by reads and writes to the 32 word data buffer array dbuf[0:31].

Writing to the dbuf_rindex register triggers reading a chunk (128 bytes) from the DRAM into the cpu data buffer dbuf[*]. The desired address is specified as the data written during the write to dbuf_rindex. The address is composed of a 2 bit bank number and a 20 bit offset within the bank. After the data has been transferred to the cpu data buffer,

A.3.0 The Packet Header Format for Cougars

C[7:0]	CD[95:0]	Source	Dest.	Description
1000xxxx	CD[95:0]	Tiger	Cougar	This is in idle state, data is don't care. Default to all "0"s
1110-SPID	CD[95:0]	Tiger-> Cougar	Cougar ->Tiger	*C-Port Data Field Vectors 1 & 2*
1111-SPID	CD[95:0]	Tiger	Cougar -> Tiger	*C-Port command vectors 1 & 2*
1111-SPID	CD[ ] [ ]	Tiger	Cougar	[ ] **Physical Source Port ID** (SP-ID)
1111-SPID	[ ]	Cheetah	Tiger & Cougar	[ ] **Switch Fabric Source Port ID** (SFSP-ID) (from Cheetah configuration register)
1111-SPID	[ ]	Cheetah	Tiger & Cougar	[ ] **Drop Precedence (DP[ ])** (from Cheetah look up table)
1111-SPID	[ ]	Cheetah	Tiger & Cougar	[ ] **Physical Destination Port ID** (DP-ID) (from Cheetah lookup table) Note:
1111-SPID	[23:16]	Cheetah	Tiger & Cougar	[23:16] **Switch Fabric Destination Port ID** (SFDP-ID) (from Cheetah lookup table)
1111-SPID	[27:24]	Cheetah	Cougar	[26:24] **Queue Pointer (QP)** When '000' => Class[0] When '001' => Class[1] When '010' => Class[2] When '011' => Class[3] When '100' => Class[4] When '101' => Class[5] When '110' => Class[6] When '111' => Class[7] [27] **Command Type**: (default is '0') When '0', indicated field [23:12] as SFDP-ID. When '1' indicated field [23:12] as MCG-ID and field [26:24] as MC Super Group ID. (MCG stand for Multicast Group)

Exhibit A
Page 16

				Note: MCG_ID is the MCG lookup pointer of Egress Cougar. MCSG_ID is the MCG lookup pointer of Ingress Cougar.
1111-SPID		Tiger	Tiger	
1111-SPID		Tiger	Tiger	
1111-SPID		Tiger	Tiger	
1111-SPID		Tiger (Ingress Cougar) (Ingress Cougar)	Cougar (Egress Cougar) (Egress Cougar)	
1111-SPID		Tiger & Cheetah (Ingress Cougar)	Cougar (Egress Cougar)	
1111-SPID				
1111-SPID		Cheetah (Ingress Cougar)	Cougar (Egress Cougar)	
1111-SPID		Cheetah Cougar	Cougar Tiger	
1111-SPID		Tiger (RX)	Tiger (TX)	
1111-SPID		Ingress Cougar	Tiger (TX)	
1111-SPID		Tiger (Rx)	Tiger (TX)	
1111-SPID		Ingress Cougar	Egress Cougar	

Exhibit A
Page 17